REMARKS

INTRODUCTION:

In accordance with the foregoing, claims 1-10 have been amended, and new claim 11 has been added. No new matter is being presented, and approval and entry are respectfully requested.

Claims 1-11 are pending and under consideration. Reconsideration is respectfully requested.

INTERVIEW REQUEST:

Applicant respectfully requests an interview with the Examiner to advance the prosecution of this matter. Please advise Applicant's attorney when it would be convenient. Applicant's attorney is Darleen J. Stockley, who may be reached at telephone number 202-434-1536. Please call to set up a time that will be convenient for the Examiner.

ENTRY OF RESPONSE UNDER 37 C.F.R. §1.116:

Applicant requests entry of this Rule 116 Response and Request for Reconsideration because:

(a) it is believed that the amendments of claims 1-10 and addition of claim 11 put this application into condition for allowance;

(b) the amendments were not earlier presented because the Applicant believed in good faith that the cited prior art did not disclose the present invention as previously claimed;

(c) the amendments of claims 1-10 and addition of claim 11 should not entail any further search by the Examiner since no new features are being added or no new issues are being raised; and/or

(d) the amendments do not significantly alter the scope of the claims and place the application at least into a better form for appeal. No new features or new issues are being raised.

The Manual of Patent Examining Procedures sets forth in §714.12 that "[a]ny amendment that would place the case either in condition for allowance or in better form for appeal may be entered." (Underlining added for emphasis) Moreover, §714.13 sets forth that "[t]he Proposed Amendment should be given sufficient consideration to determine whether the claims are in condition for allowance and/or whether the issues on appeal are simplified." The Manual of Patent Examining Procedures further articulates that the reason for any non-entry should be explained expressly in the Advisory Action.

REJECTION UNDER 35 U.S.C. §103:

In the Office Action, at pages 2-9, numbered paragraphs 3-6, claims 1-10 were rejected under 35 U.S.C. §103(a) as being unpatentable over Glass et al. (US006629128B1; hereafter, Glass) in view of Howes et al. (US006324177B1; hereafter, Howes) and further in view of Dugan et al. (US006425005B1; hereafter, Dugan). The reasons for the rejection are set forth in the Office Action and therefore not repeated. The rejection is traversed and reconsideration is requested.

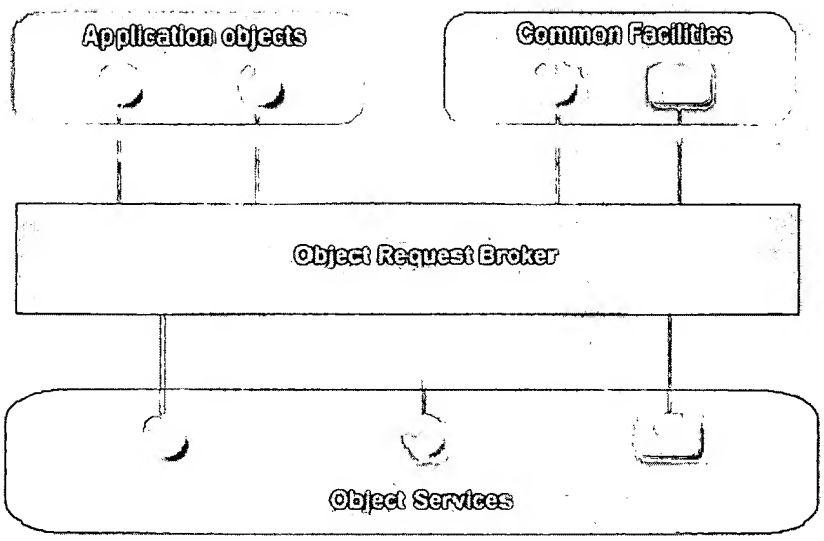
The Examiner admits that Glass does not explicitly teach:

- a request receiving unit which receives a request from an apportioning server, initially sent by a client connected via a network, to acquire an object reference for receiving a distribution of a naming service in CORBA,
- wherein the apportioning server has determined whether an arrival address is an apportioning IP address, and if the result is negative, establishes a connection with the arrival IP address, and if the result is positive, distributes a load to a server having a lightest load in comparison with other servers.

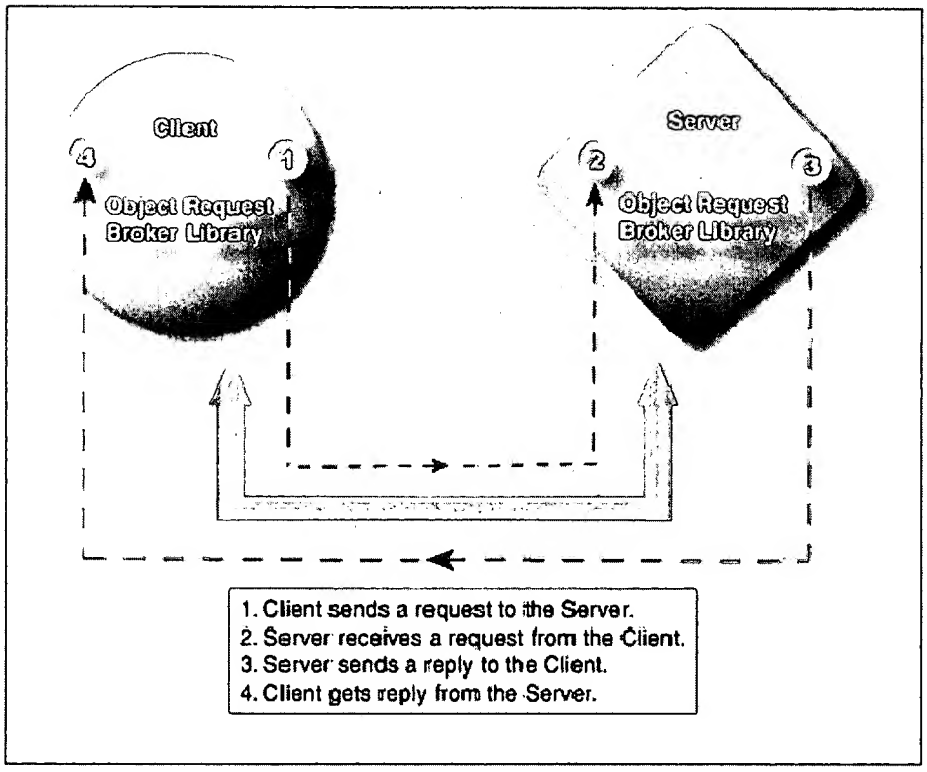
It is respectfully submitted that it may be helpful to describe CORBA, which is utilized in the present invention, more specifically. CORBA is essentially a design specification for an Object Request Broker (ORB), where an ORB provides the mechanism required for distributed objects to communicate with one another, whether locally or on remote devices, written in different languages, or at different locations on a network.

As noted at www.ois.com/resources/corb-2.asp, repeated below for the Examiner's convenience:

> CORBA is often described as a "software bus" because it is a software-based communications interface through which objects are located and accessed. The illustration below identifies the primary components seen within a CORBA implementation.



Data communication from client to server is accomplished through a well-defined object-oriented interface. The Object Request Broker (ORB) determines the location of the target object, sends a request to that object, and returns any response back to the caller. Through this object-oriented technology, developers can take advantage of features such as inheritance, encapsulation, polymorphism, and runtime dynamic binding. These features allow applications to be changed, modified and re-used with minimal changes to the parent interface. The illustration below identifies how a client sends a request to a server through the ORB:



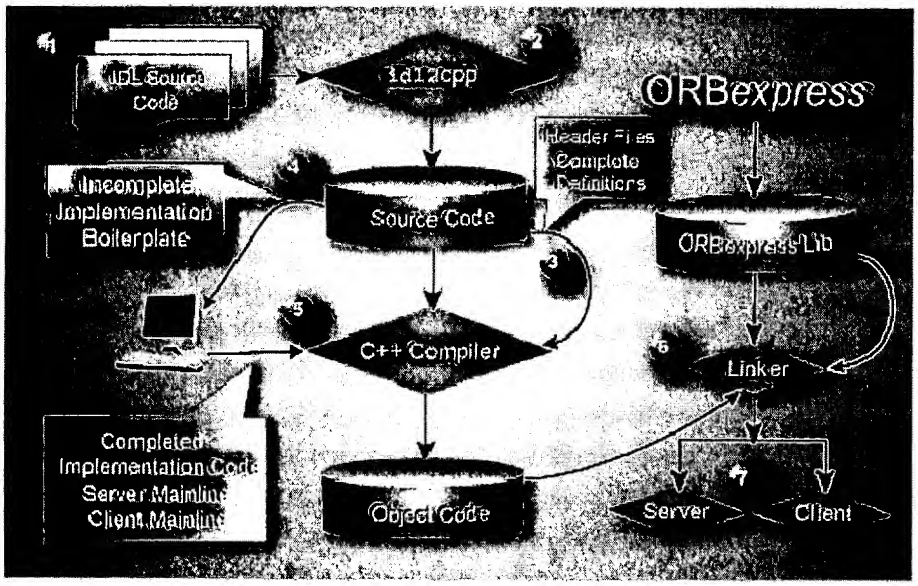
Interface Definition Library

A cornerstone of the CORBA standards is the Interface Definition Library, or IDL. IDL is the OMG standard for defining language-neutral APIs and provides the platform-

independent delineation of the interfaces of distributed objects. The ability of the CORBA environments to provide consistency between clients and servers in heterogeneous environments begins with a standardized definition of the data and operations constituting the client/server interface. This standardization mechanism is the IDL, and is used by CORBA to describe the interfaces of objects.

IDL defines the modules, interfaces and operations for the applications and is not considered a programming language. The various programming languages, such as Ada, C++, or Java, supply the implementation of the interface via standardized IDL mappings.

CORBA Development Process
The basic steps for CORBA development can be seen in the illustration below. This illustration provides an overview of how the IDL is translated to the corresponding language (in this example, C++), mapped to the source code, compiled, and then linked with the ORB library, resulting in the client and server implementation.



The basics steps for CORBA development include:

❶*Create the IDL to define the application interfaces*
The IDL provides the operating system and programming language independent interfaces to all services and components that are linked to the ORB. The IDL specifies a description of any services a server component exposes to the client. The term "IDL Compiler" is often used, but the IDL is actually translated into a programming language.

❷*Translate the IDL*
An IDL translator typically generates two cooperative parts for the client and server implementation, stub code and skeleton code. The stub code generated for the interface classes is associated with a client application and provides the user with a well-defined API (Application Programming Interface). In this example, the IDL is translated into C++.

❸*Compile the interface files*
Once the IDL is translated into the appropriate language, C++ in this example, these interface files are compiled and prepared for the object implementation.

❹*Complete the implementation*
If the implementation classes are incomplete, the spec and header files and complete

bodies and definitions need to be modified before passing through to be compiled. The output is a complete client/server implementation.

❺*Compile the Implementation*
Once the implementation class is complete, the client interfaces are ready to be used in the client application and can be immediately incorporated into the client process. This client process is responsible for obtaining an object reference to a specific object, allowing the client to make requests to that object in the form of a method call on its generated API.

❻*Link the Application*
Once all the object code from steps three and five have been compiled, the object implementation classes need to be linked to the C++ linker. Once linked to the ORB library, in this example, ORBexpress, two executable operations are created, one for the Client and one for the Server.

❼*Run the Client and Server*
The development process is now complete and the Client will now communicate with the Server. The Server uses the object implementation classes allowing it to communicate with the objects created by the Client requests.

In its simplest form, the Server must perform the following:

- Create the required objects.
- Notify the CORBA environment that it is ready to receive client requests.
- Process client requests by dispatching the appropriate servant.

In addition, it is known to those skilled in the art that, when using CORBA, the ORB allows a client to transparently invoke operations on a server object. In order to achieve this location transparency, CORBA provides a naming service. In order for a server object to make its operations available to clients, it must register an object reference as well as a unique name with the naming service. The naming service stores object references in a tree-like structure similar to that of a file system. When a client wishes to avail itself of the services of a particular server object, it can query the naming service and obtain an object reference for the named object. The client can then invoke operations on that object using the object reference. The naming service is similar to the telephone white pages directory.

Hence, CORBA objects can be located anywhere on a network, can interoperate with objects on other platforms, and can be written in any programming language for which there is a mapping from OMG IDL to that language. Independent claims 1, 5, 6, 7, and 8 have been amended to clarify differences in the present invention, which utilizes CORBA, and the invention of Howes, which does not utilize CORBA.

Thus, it is respectfully submitted that the present invention utilizes CORBA (see, for example, page 1, lines 8-17, and claim 1). In contrast, Howes does not utilize CORBA - when a word search is done on the Howes patent, there is no mention of CORBA. instead of utilizing

the CORBA system, Howes teaches, col. 3, lines 13-42: "Incoming SYN packets are analyzed to determine the source IP address of new connection requests. The source IP address is used to determine which web server is assigned to handle the connection requested by the SYN packet. ... In one embodiment, a method for assigning an incoming connection to a server includes defining a client specific virtual machine object instance that is associated with a designated client IP address. An incoming packet is received that is associated with a new connection. The incoming packet has a packet source IP address, a packet source port number, a packet destination IP address, and a packet destination port number. A client specific virtual machine object instance is selected that is associated with the designated client IP address when the packet source IP address matches the designated client IP address. The packet destination IP address is translated to a physical machine IP address that is designated by the client specific virtual machine instance when the packet source IP address matches the designated client IP address so that the client specific virtual machine instance is selected when the designated client IP address matches the packet source IP address and the packet is routed to the physical machine that is designated by the client specific virtual machine instance." (emphasis added) Thus, Howe teaches use of a client specific virtual machine object, in contrast to the CORBA object used by the present invention.

Thus, the Examiner's contention that Howes teaches "a request receiving unit which receives a request from an apportioning server, initially sent by a client connected via a network, to acquire an object reference for receiving a distribution of a naming service in CORBA" is submitted to be incorrect, and Howes should not be combined with Glass, and Howe teaches away from the present claimed invention.

Dugan teaches using an Intelligent Distributed Network Architecture (IDNA) (see Dugan, col. 8, lines 3-18) that implements intelligent call processors in place of monolithic switches (see Dugan, col. 9, lines 4-7). Dugan teaches, col. 9, lines 8-48:

> In the case of applications that require more processing power, multi-processing allows the use of less expensive processors to optimize the price/performance ratio for call processing. In other applications, it may be advantageous, necessary or more cost effective to use more powerful machines, such as minicomputers, with higher processing rates.
>
> The ICP 172 may, as noted above, comprise a cluster of general purpose computers operating, for example, on a UNIX or Windows NT operating system. For example, in a large application, supporting up to 100,000 ports on a single Resource Complex, the ICP 172 may consist of sixteen (16) 32 bit processors operating at 333 MHz in a Symmetric Multi-Processor cluster. The processors could, for example, be divided into four separate servers with four processors each. The individual processors would be connected with a System Area Network ("SAN") or other clustering technology. The processor cluster could share access to Redundant Array of Independent Disks ("RAID") modular data storage devices. Shared storage may be adjusted by adding or removing the modular

disk storage devices. The servers in the clusters would preferably share redundant links to the RC 180 (FIG. 1).

As illustrated and like the "plug and play" feature of personal computers, the ICP software architecture is an open processing model that allows the interchangeability of: (1) management software; (2) ICP applications; (3) computing hardware and software; (4) resource complex components; and even (5) service architecture and processing. Such a generic architecture reduces maintenance costs due to standardization and provides the benefits derived from economies of scale.

Thus, the present invention enables the partitioning of development work and the use of modular tools that result in faster development and implementation of services. Moreover, the use of and the relevant aspects of service management are within the control of the network operator on an as required basis as opposed to the constraints imposed by fixed messaging protocol or a particular combination of hardware and software supplied by a given manufacturer.

Hence, it is respectfully submitted that Dugan does not teach

- a request receiving unit which receives a request from an apportioning server, initially sent by a client connected via a network, to acquire an object reference for receiving a distribution of a naming service in CORBA,
- wherein the apportioning server has determined whether an arrival address is an apportioning IP address, and if the result is negative, establishes a connection with the arrival IP address, and if the result is positive, distributes a load to a server having a lightest load in comparison with other servers,

which are also not taught or suggested by Glass.

Hence, it is respectfully submitted that, even if combined, Glass and Dugan do not teach or suggest amended independent claims 1, 5, 6 and/or 8 of the present invention. Since Howes teaches away from the present invention by implementing a non-CORBA technique, even if combined, Glass, Dugan, and Howes fail to teach or suggest claims 1, 5, 6 and/or 8 of the present invention.

Thus, it is respectfully submitted that amended independent claims 1, 5, 6, 7 and 8 of the present invention are patentable under 35 U.S.C. §103(a) over Glass et al. (US006629128B1) in view of Howes et al. (US006324177B1) and further in view of Dugan et al. (US006425005B1). Since claims 2-4, 9 and 10 depend from amended independent claims 1 and 5, respectively, claims 2-4, 9 and 10 are patentable under 35 U.S.C. §103(a) over Glass et al. (US006629128B1) in view of Howes et al. (US006324177B1) and further in view of Dugan et al. (US006425005B1) for at least the reasons that amended independent claims 1 and 5 are patentable under 35 U.S.C. §103(a) over Glass et al. (US006629128B1) in view of Howes et al. (US006324177B1) and further in view of Dugan et al. (US006425005B1).

NEW CLAIM:

New claim 11 recites that the features of the present invention include the CORBA object

reference generating device according to claim 2, wherein the arrival address information is the apportioning IP address. The new claim is based on the description of the specification, page 18, lines 13-20. Nothing in the prior art teaches or suggests such. It is submitted that this new claim distinguishes over the prior art.

EXAMINER'S RESPONSE TO ARGUMENTS:

In the Office Action, at pages 9-12, numbered paragraphs 7-10, the Examiner submitted his response to Applicant's arguments filed December 21, 2006.

In view of the above amendments and arguments, it is respectfully submitted that the Examiner's concerns have been overcome.

CONCLUSION:

In accordance with the foregoing, it is respectfully submitted that all outstanding objections and rejections have been overcome and/or rendered moot, and further, that all pending claims patentably distinguish over the prior art. Thus, there being no further outstanding objections or rejections, the application is submitted as being in condition for allowance which action is earnestly solicited. At a minimum, this Amendment should be entered at least for purposes of Appeal as it either clarifies and/or narrows the issues for consideration by the Board.

If the Examiner has any remaining issues to be addressed, it is believed that prosecution can be expedited and possibly concluded by the Examiner contacting the undersigned attorney for a telephone interview to discuss any such remaining issues.

If there are any underpayments or overpayments of fees associated with the filing of this Amendment, please charge and/or credit the same to our Deposit Account No. 19-3935.

Respectfully submitted,

STAAS & HALSEY LLP

Date: June 19, 2007

By: Darleen J. Stockley
Darleen J. Stockley
Registration No. 34,257

1201 New York Avenue, N.W.
Suite 700
Washington, D.C. 20005
Telephone: (202) 434-1500
Facsimile: (202) 434-1501